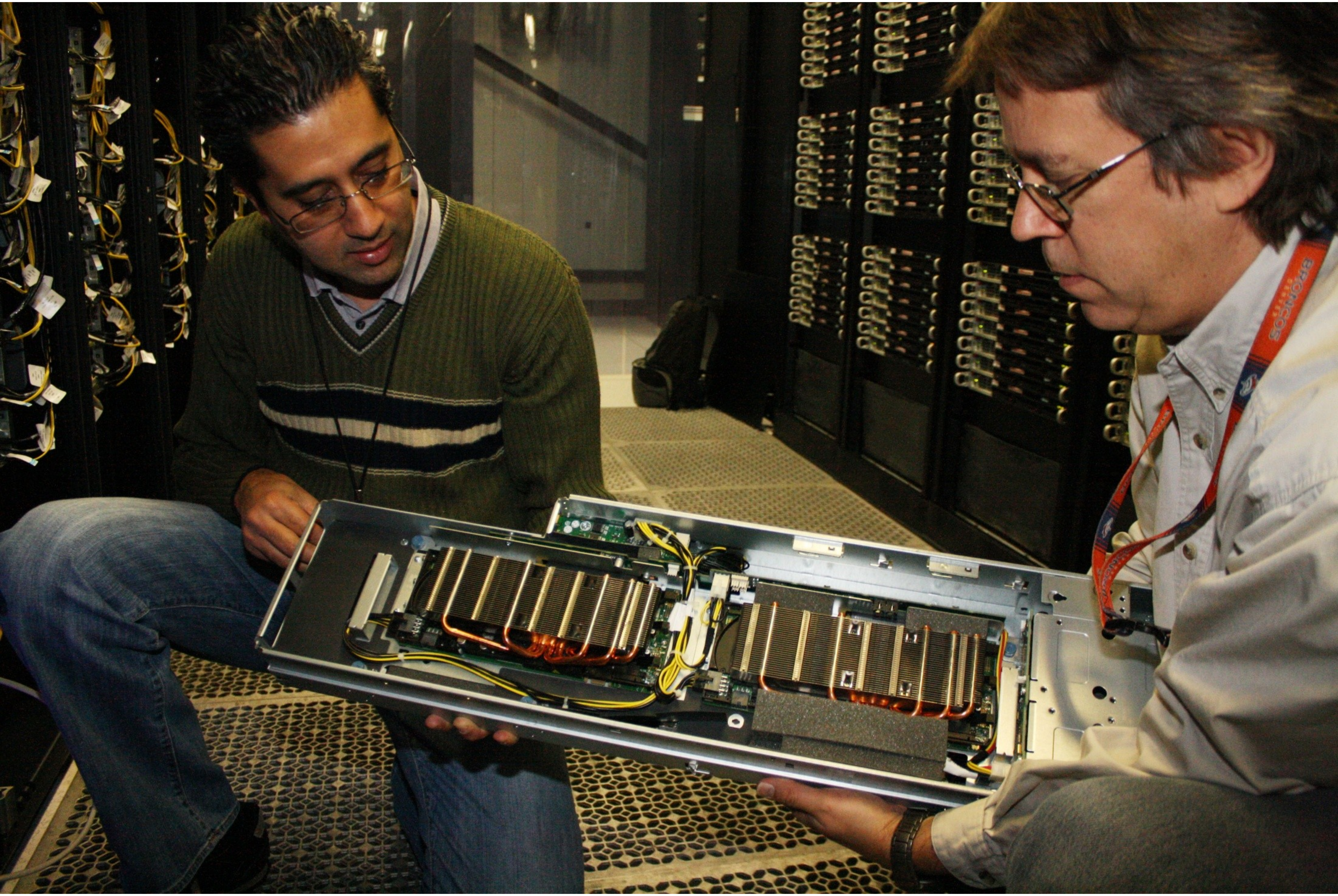
GPU Use for Lattice QCD and Other Calculations

Don Holmgren

CS/SC/SCF/HPC

All Experimenters Meeting, February 20, 2012



BRONCOS

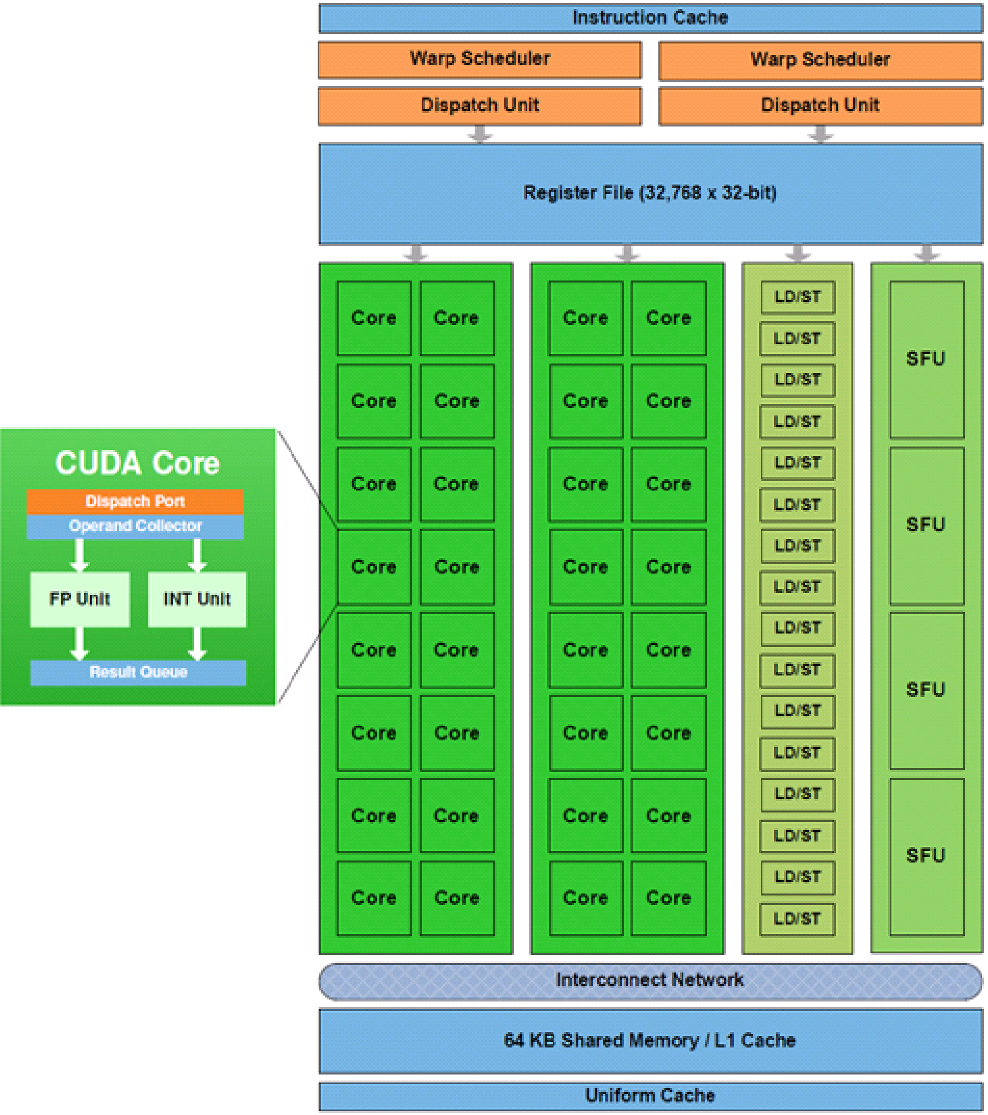
Outline

- Introduction to GPUs
- Lattice QCD and GPUs
- Other GPU Work at Fermilab:
  - Network monitoring
  - Accelerator Modeling
  - Event Generation

Introduction to GPUs

- All current desktops, laptops, and servers have a GPU (Graphics Processing Unit), and the majority use silicon from either Nvidia or ATI (owned by AMD). This talk focuses on Nvidia.
- Starting in the late 90’s, GPUs in general became “functionally complete” and so could support general computation
- Early adopters used shading languages such as DirectX, OpenGL, or Cg to perform computations, mapping mathematical operations onto graphical operations such as vertex rotation. This practice was called “GPGPU” (General Purpose computations on GPU).
- In 2006 Nvidia produced a new series of GPUs (G80-family) that supported a programming API called CUDA (Compute Unified Device Architecture) which allowed programmers to use C (and later Fortran, C++, Java, Python…)

The “Fermi” GPU



Fermi Streaming Multiprocessor (SM)

- Nvidia’s latest GPU is called the “Fermi”
- Each Fermi has a number of Streaming Multiprocessors (SM)
- Each SM has 32 CUDA cores
- Each CUDA core can retire a floating point or integer instruction per clock
- An SM also has Load/Store units, and Special Function Units (for transcendental operations)
- Full IEEE 754-2008 32-bit (single) and 64-bit (double) precision floating point
- Groups of cores can execute the same program (“kernel”) in parallel against different sets of data
- Register and shared memory are fast but local. SM’s can also access slower global memory on the GPU card. Host memory can be accessed over PCIe bus.

- GPUs are used as accelerators, not as stand-alone computers
  - They do not run an operating system
  - A CUDA program launches computational kernels that may run asynchronously with the program executing on the host CPU
  - The speedup of a given program will depend on the relative fractions of time spent in the GPU-parallelizable and serial (CPU-only) portions

Lattice QCD

USQCD

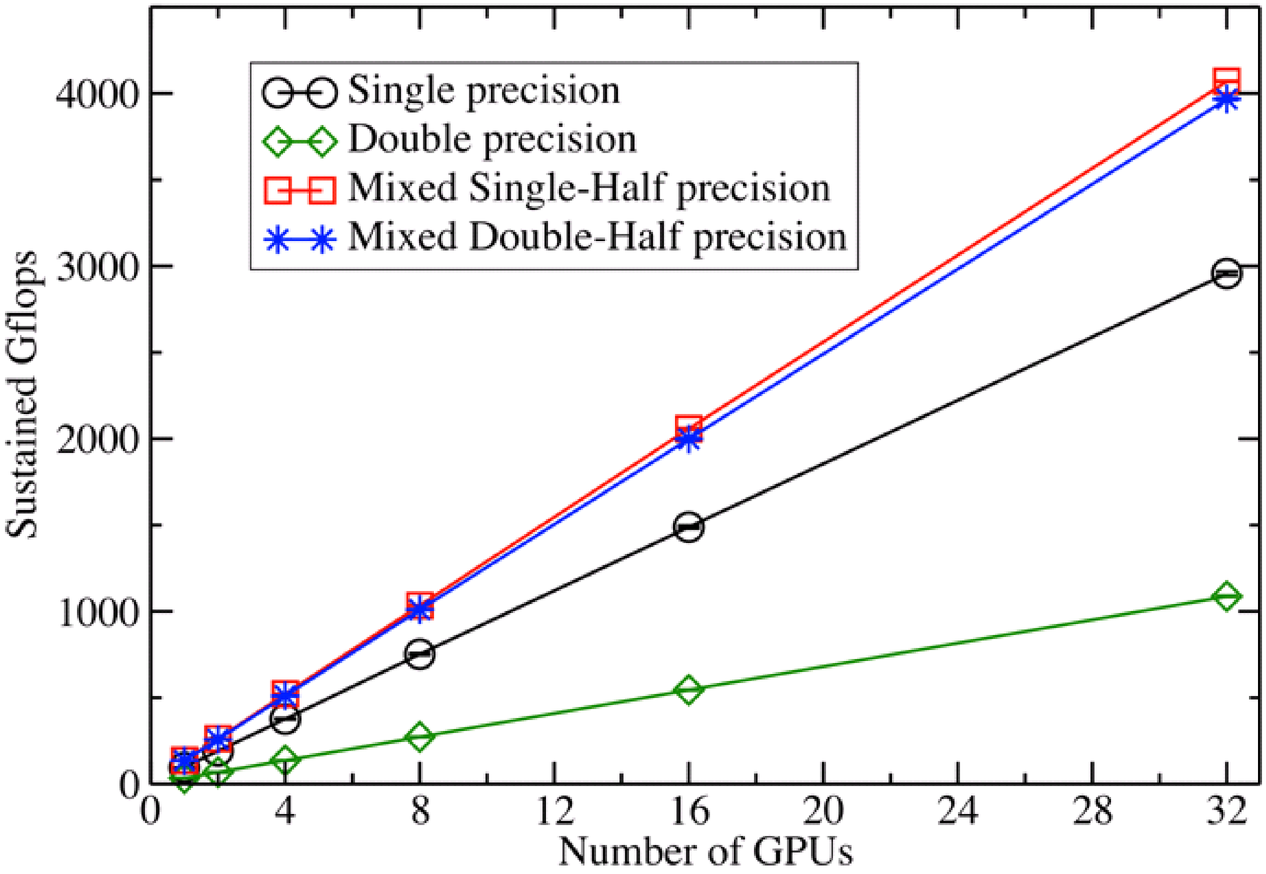
US Lattice Quantum Chromodynamics

- The High Performance Computing Department designs, procures, and operates computer clusters dedicated to Lattice QCD computations as part of the DOE SC LQCD-ext project
  - $18.15M over 5 years (FY10-FY14), $14M of the total at FNAL, a continuation of the 4-year, $9.2M DOE SC LQCD project (FY06-FY09), funded by DOE NP and HEP
  - Three labs: FNAL, Jefferson Lab, Brookhaven
  - Fermilab personnel: Bill Boroski (Contractor Project Manager), Bakul Banerjee (Associate Contractor Project Manager), Jim Simone (Dept Head), Amitoj Singh (Deputy Dept Head), Don Holmgren, Nirmal Seenu, Bob Forster, Rick van Conant, Ken Schumacher
- DOE SC (HEP, NP, ASCR) since 2001 has funded LQCD software development through the SciDAC, SciDAC-2, and (we hope) SciDAC-3
- USQCD, a collaboration of most US lattice theorists, allocates time on the LQCD-ext systems. Paul Mackenzie is the Chair of the USQCD Executive Committee. For more information, see http://www.usqcd.org/

GPU-Accelerated LQCD

- Lattice theorists in Europe started using GPUs in 2005, coding in OpenGL with Cg
- Barros *et al*. (http://arxiv.org/abs/0810.5365) at Boston U. implemented a Wilson-Dirac operator in CUDA in 2007 with about a 10x speedup over conventional x86 hardware
- The Boston U. work evolved under the SciDAC-2 program into *QUDA* (http://lattice.github.com/quda/)
  - Initially a single GPU code (http://arxiv.org/abs/0911.3191) for Wilson-Dirac
  - Mixed-precision algorithms were adapted and exploited to improve performance
  - Reduced representations were used to minimize memory operations and increase performance
- *QUDA* has since further evolved:
  - Support for other actions: Clover, twisted-mass, asqtad (with Indiana U. and NCSA), domain wall
  - Multi GPU support (Boston U. + JLab, http://arxiv.org/abs/1011.0024)
- LQCD SC11 GPU paper:
  - "Scaling Lattice QCD beyond 100 GPUs" http://arxiv.org/abs/1109.2935

Weak Scaling Results with Mixed Precision Optimizations



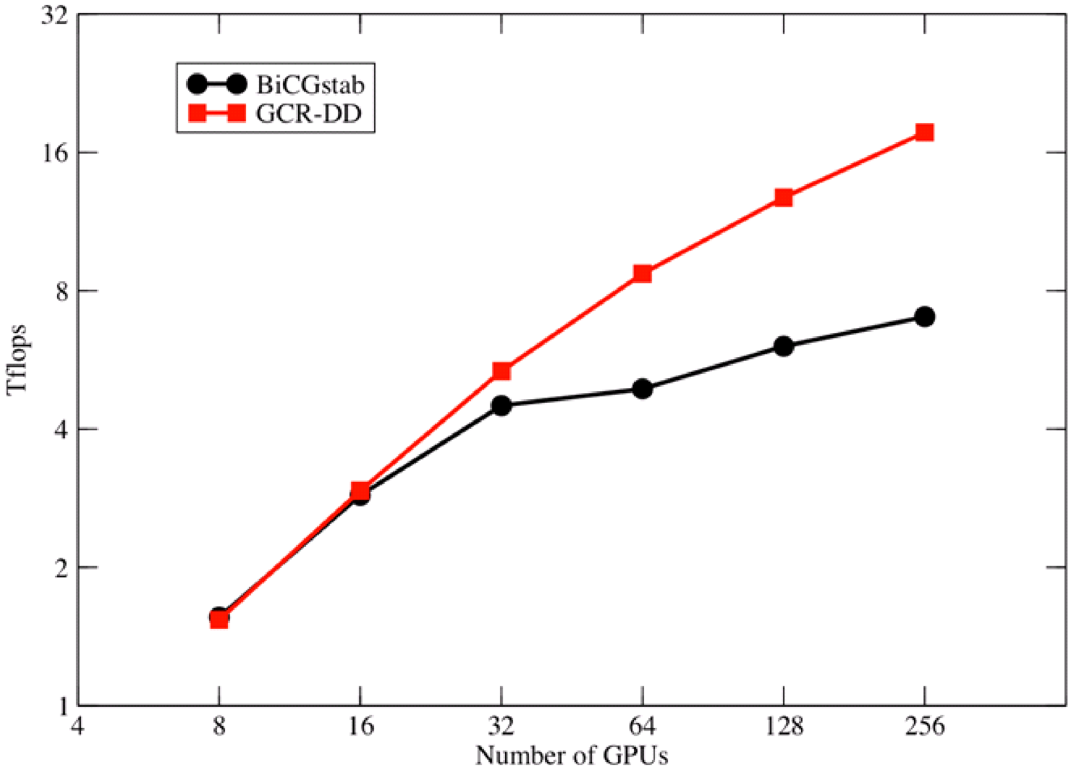
Gflops shown are *effective* (*i.e.* they are the equivalent Gflops that would be observed on conventional CPUs for the same rate of work on the same problem.

Data are from the Jefferson Lab “9g” cluster, using GTX-285 GPUs.

These optimizations (mixed precision, reliable updates) are also relevant to conventional processors.

Our fastest conventional cluster (Ds) sustains about 50 Gflops per 32-core node

Strong Scaling Performance

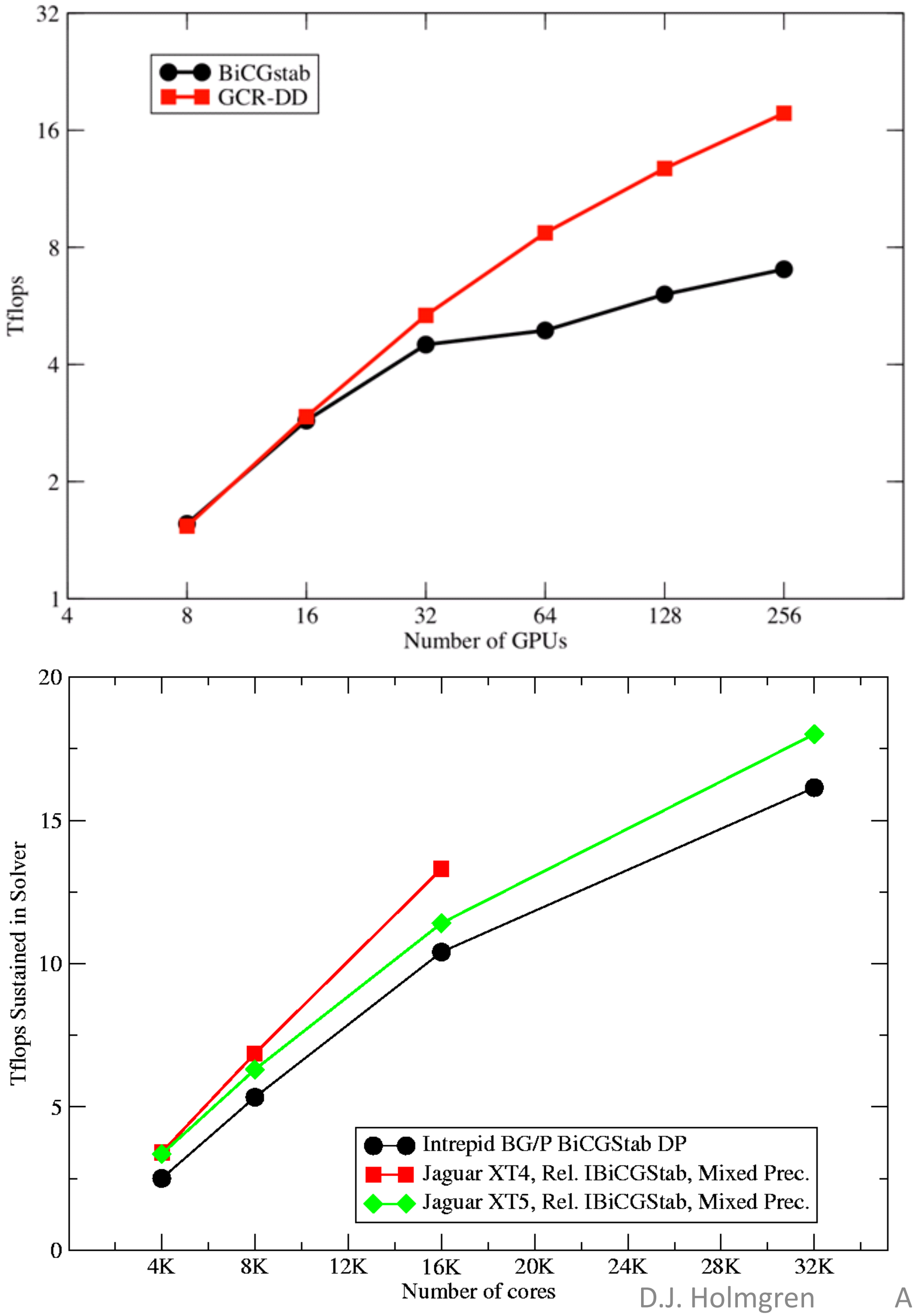


Data taken on the Edge GPU cluster at LLNL (nVidia Tesla M2050 GPUs).

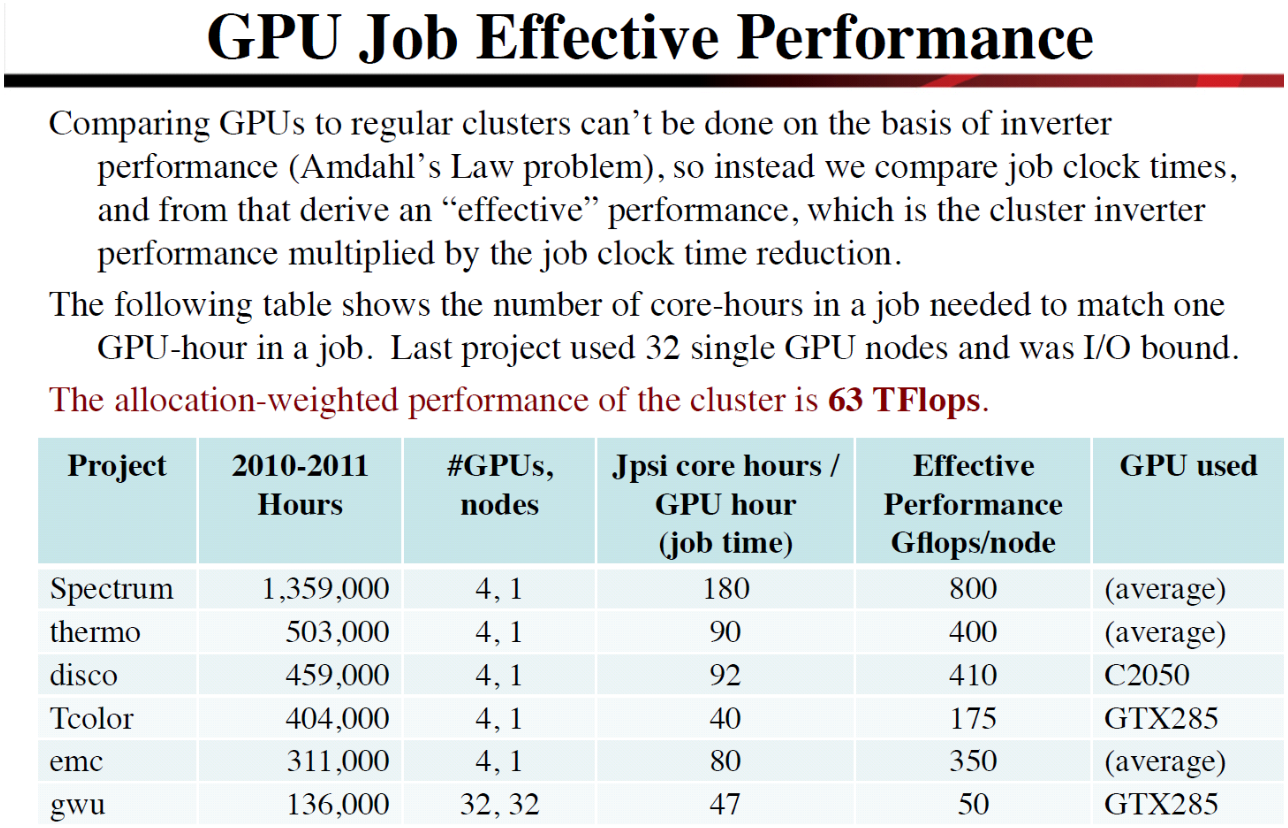
TFlops are effective.

Data are for a Wilson-Clover problem of size 32^3 x 256.

The “GCR-DD” data use a communication-minimizing solver that relies on additive Schwarz pre-conditioners to perform domain decomposition.



Strong scaling results for the same problem (Wilson-Clover 32^3 x 256) on the Edge GPU cluster at LLNL (top), and on the ORNL XT4 and XT5 and ANL Intrepid BG/P (bottom).



GPU Job Effective Performance

Comparing GPUs to regular clusters can't be done on the basis of inverter performance (Amdahl's Law problem), so instead we compare job clock times, and from that derive an "effective" performance, which is the cluster inverter performance multiplied by the job clock time reduction.

The following table shows the number of core-hours in a job needed to match one GPU-hour in a job. Last project used 32 single GPU nodes and was I/O bound.

The allocation-weighted performance of the cluster is **63 TFlops**.

Project	2010-2011 Hours	#GPUs, nodes	Jpsi core hours / GPU hour (job time)	Effective Performance Gflops/node	GPU used
Spectrum	1,359,000	4, 1	180	800	(average)
thermo	503,000	4, 1	90	400	(average)
disco	459,000	4, 1	92	410	C2050
Tcolor	404,000	4, 1	40	175	GTX285
emc	311,000	4, 1	80	350	(average)
gwu	136,000	32, 32	47	50	GTX285

LQCD performance strongly varies with the problem and what parts of the particular code have been re-written in QUDA or CUDA.

The projects above were running on USQCD GPU hardware at Jefferson Lab last year. "Jpsi core hours" are the USQCD allocation unit. Acceleration ranges from 5.9:1 to 22:1 (comparing various single GPUs to performance of 8-core 2.1 GHz AMD Barcelona nodes).

Fermilab USQCD Hardware

- "D_s"
  - 13472 cores in 421 quad-socket eight-core AMD Magny-Cours servers, QDR Infiniband
  - 21.5 TF sustained for LQCD (Linpack TF about 80 TF)
- "J/Psi"
  - 6848 cores in 856 dual-socket quad-core AMD Barcelona servers, DDR Infiniband, plus 8 servers with dual nVidia C1060 GPUs
  - 8.4 TF sustained for LQCD, Top500 #110 June 2009
- GPU-accelerated cluster "D_sg"
  - 152 nVidia M2050 Fermi GPUs in 76 dual-socket quad-core Intel Westmere servers, interconnected with QDR Infiniband
  - Delivered in January, in "friendly-user" mode now
- Lustre file system
  - 543 TB
  - 114 OSTs
  - ~ 1600 clients

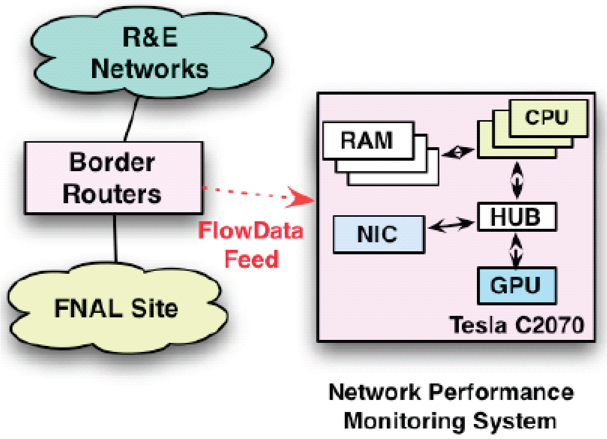
“D_sg” Details

- Vendor: Hewlett-Packard
- Cost: $525K
- Hosts:
  - Dual socket, 4 cores/socket, 2.53 GHz, Intel “Westmere” processors
  - 48 GBytes memory per node, 500 GB local disk
  - 16 nodes per rack, 12 KW maximum per rack
  - 2 GPUs per host, 76 hosts, 152 GPUs total
- GPU’s:
  - Nvidia Tesla M2050 (“Fermi”)
  - 3 GBytes memory, ECC-capable, hardware double precision
  - 448 CUDA cores per GPU, 1 Tflop/sec peak single precision performance
- Networking:
  - Quad data rate Infiniband (Mellanox)
  - 40 Gbits/sec/direction signaling rate (32 Gbits/sec data rate)

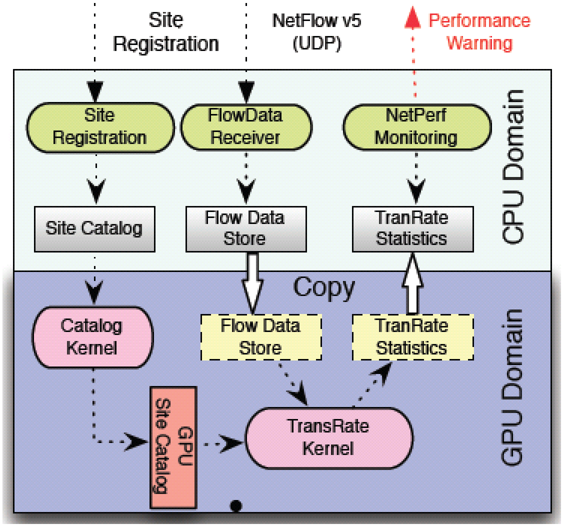
Network Monitoring

G-NetMon: A GPU-accelerated Network Performance Monitoring System for Large Scale Scientific Collaborations

W. Wu, P. DeMar, D. Holmgren, A. Singh, R. Pordes
Email: {wenji, demar, djholm, amitoj,ruth}@fnal.gov



G-NetMon Deployment



G-NetMon Architecture

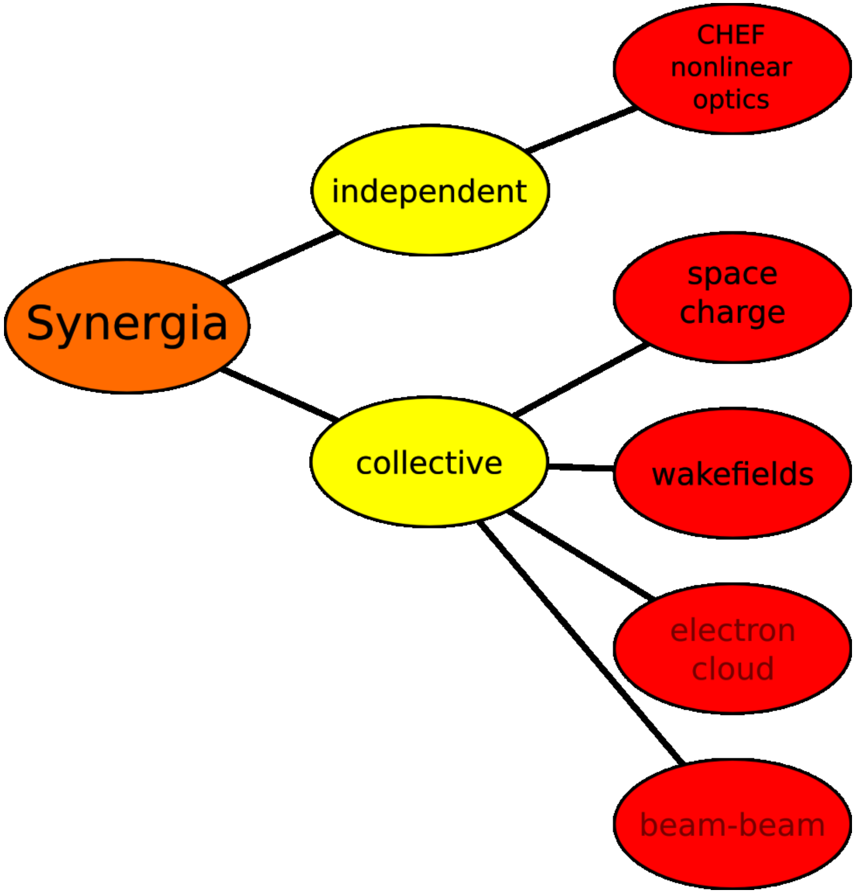
- A GPU-accelerated network performance monitoring system.
- G-NetMon exploits the inherent data parallelism that exists within network flow data and uses a GPU to rapidly calculate transfer rates between Fermilab and collaboration sites in near real time.
- G-NetMon can rapidly detect sub-optimal bulk data movements.

FERMILAB-CONF-11-362-CD
http://cd-docdb.fnal.gov/cgi-bin/ShowDocument?docid=4354

Accelerator Modeling and Simulation with GPUs (*SynergiaGPU*)

Synergia

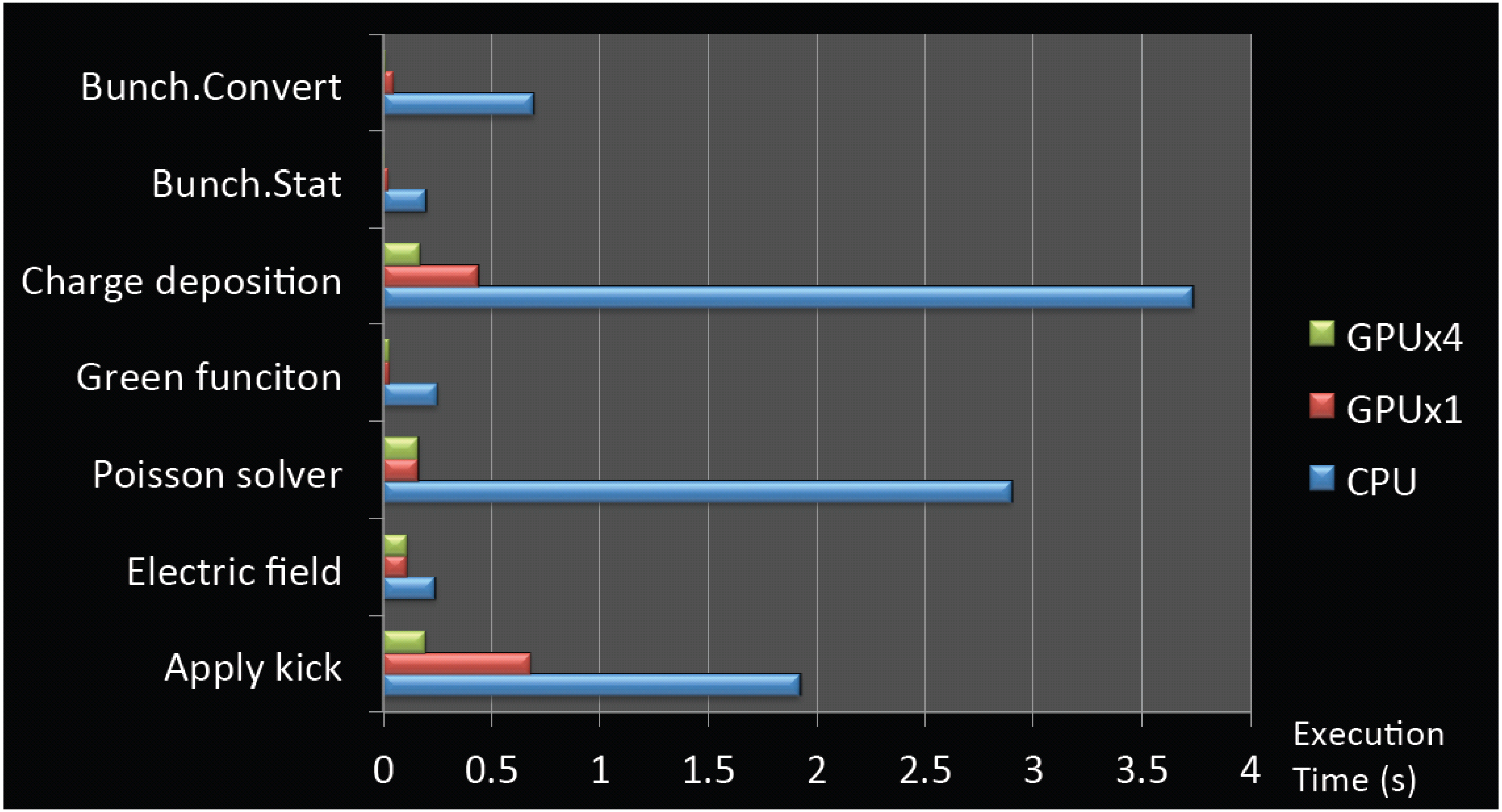
- Beam-dynamics framework developed at Fermilab
- Mixed C++ and Python
- Designed for MPI-based parallel computations
  - Desktops (laptops)
  - Clusters
  - Supercomputers



https://compacc.fnal.gov/projects/wiki/synergia2

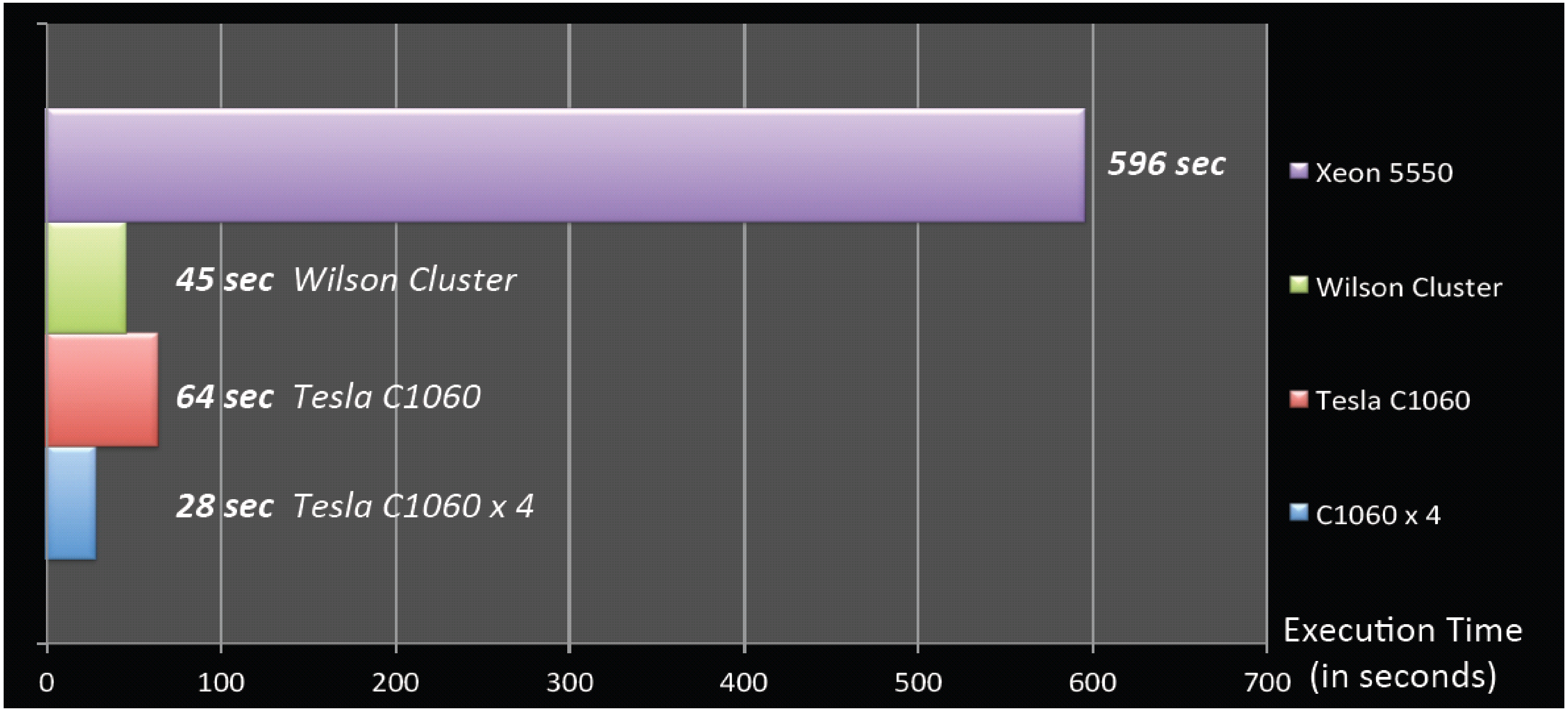
Performance Comparison for Breakdown Operations

- 1. Intel Xeon X5550, single process @ 2.67GHz;
- 2. NVidia Tesla C1060, 30 streaming multi-processors @ 1.30GHz in a single GPU
- 3. Nvidia Tesla C1060 x 4

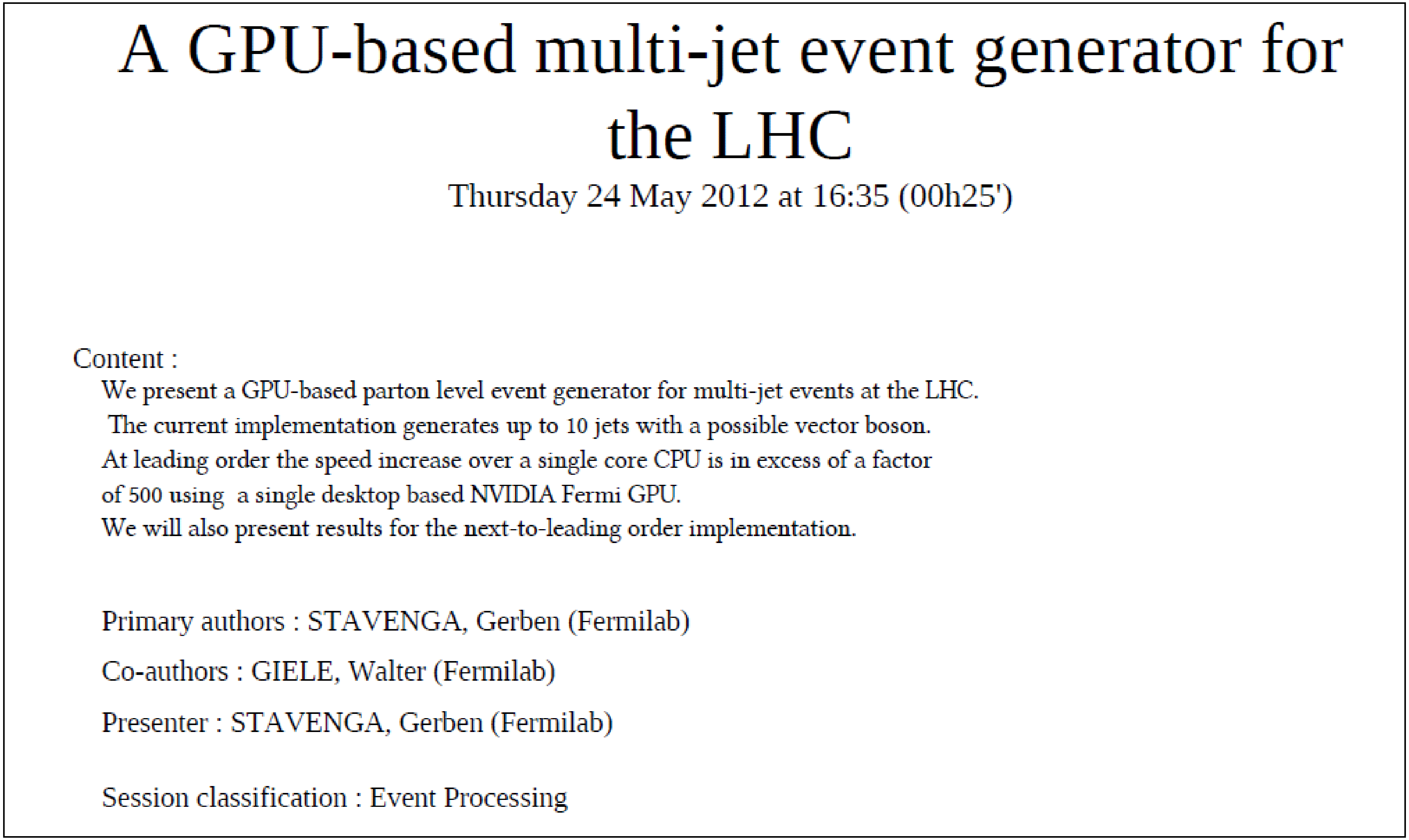


Overall Performance Comparison

- Comparison systems:
  - **1. Intel Xeon X5550, single process @ 2.67GHz;**
  - **2. Fermilab Wilson Cluster, dual Xeon X5650 2.67GHz nodes. 16 nodes / 128 cores used**
  - **3. NVidia Tesla C1060, 30 streaming multi-processors @ 1.30GHz in a single GPU**
  - **4. Nvidia Tesla C1060 x 4**



Event Generation



A GPU-based multi-jet event generator for the LHC

Thursday 24 May 2012 at 16:35 (00h25')

Content :

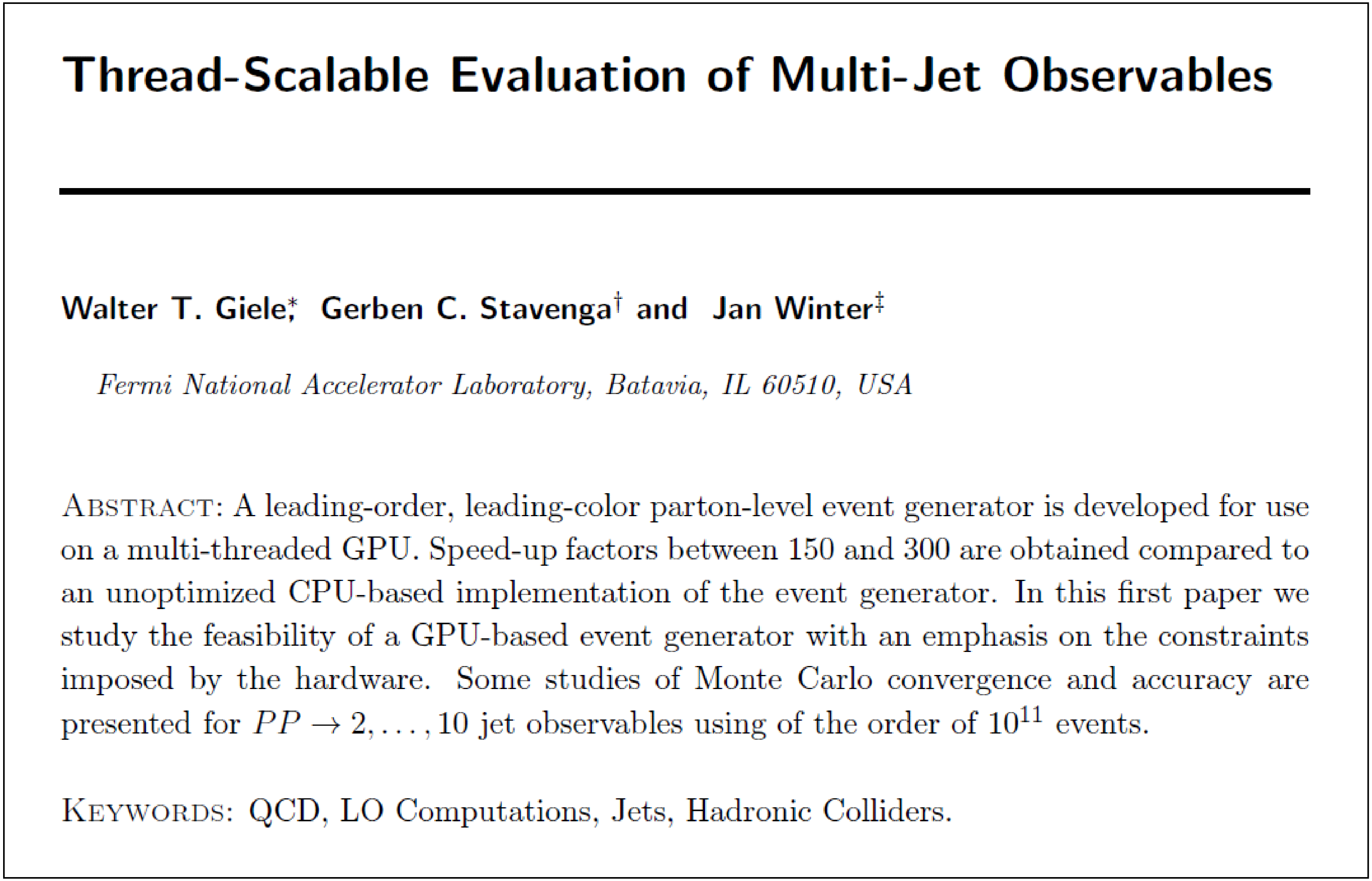
We present a GPU-based parton level event generator for multi-jet events at the LHC. The current implementation generates up to 10 jets with a possible vector boson. At leading order the speed increase over a single core CPU is in excess of a factor of 500 using a single desktop based NVIDIA Fermi GPU. We will also present results for the next-to-leading order implementation.

Primary authors : STAVENGA, Gerben (Fermilab)

Co-authors : GIELE, Walter (Fermilab)

Presenter : STAVENGA, Gerben (Fermilab)

Session classification : Event Processing



Thread-Scalable Evaluation of Multi-Jet Observables

Walter T. Giele[*], Gerben C. Stavenga[†] and Jan Winter[‡]

Fermi National Accelerator Laboratory, Batavia, IL 60510, USA

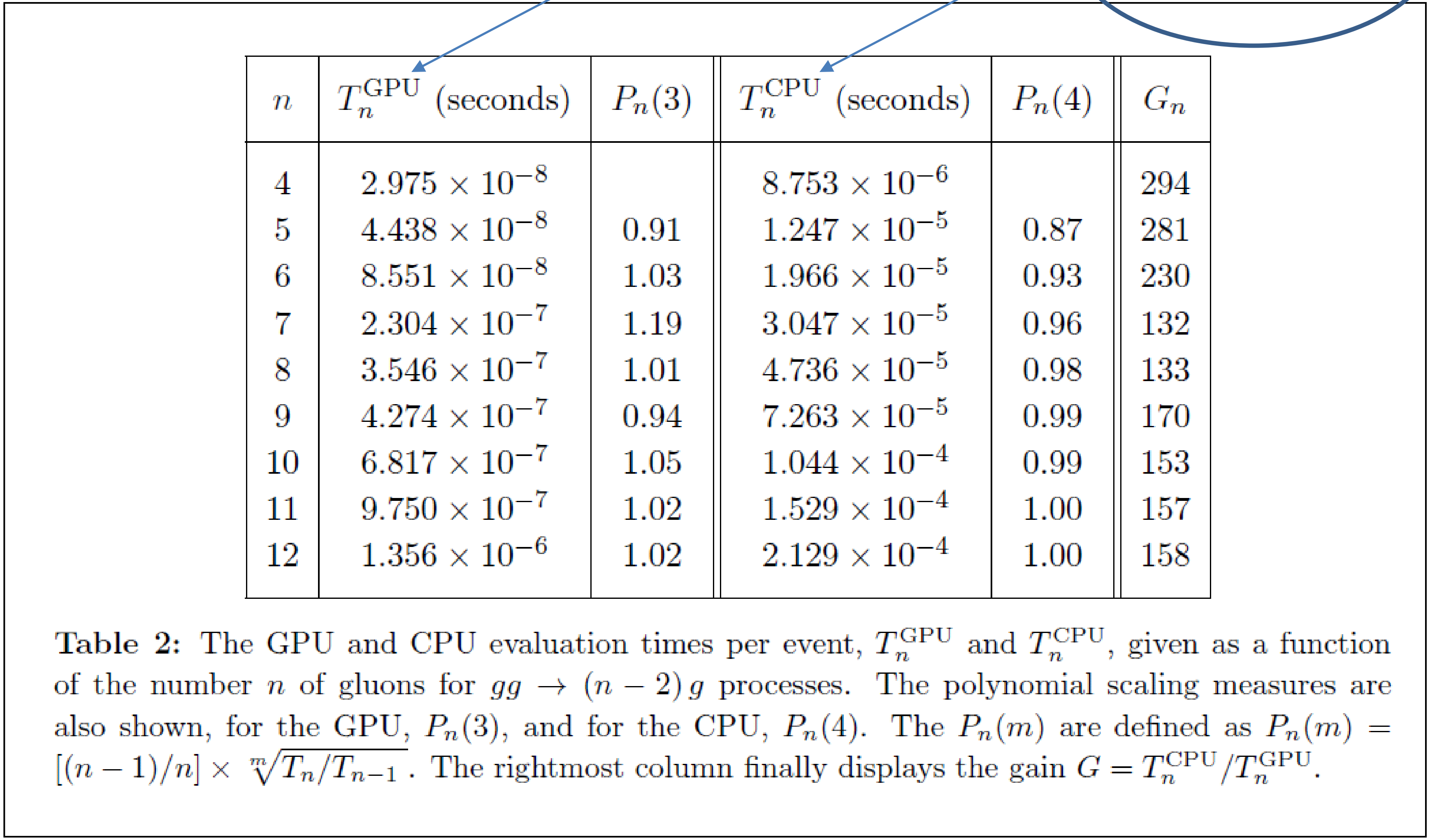
ABSTRACT: A leading-order, leading-color parton-level event generator is developed for use on a multi-threaded GPU. Speed-up factors between 150 and 300 are obtained compared to an unoptimized CPU-based implementation of the event generator. In this first paper we study the feasibility of a GPU-based event generator with an emphasis on the constraints imposed by the hardware. Some studies of Monte Carlo convergence and accuracy are presented for $PP \to 2, \ldots, 10$ jet observables using of the order of 10^{11} events.

KEYWORDS: QCD, LO Computations, Jets, Hadronic Colliders.

arXiv:1002.3446v1 [hep-ph] 18 Feb 2010

Nvidia C1060

AMD Phenom II X4 940 CPU (3 GHz)



n	T_n^{GPU} (seconds)	$P_n(3)$	T_n^{CPU} (seconds)	$P_n(4)$	G_n
4	2.975×10^{-8}		8.753×10^{-6}		294
5	4.438×10^{-8}	0.91	1.247×10^{-5}	0.87	281
6	8.551×10^{-8}	1.03	1.966×10^{-5}	0.93	230
7	2.304×10^{-7}	1.19	3.047×10^{-5}	0.96	132
8	3.546×10^{-7}	1.01	4.736×10^{-5}	0.98	133
9	4.274×10^{-7}	0.94	7.263×10^{-5}	0.99	170
10	6.817×10^{-7}	1.05	1.044×10^{-4}	0.99	153
11	9.750×10^{-7}	1.02	1.529×10^{-4}	1.00	157
12	1.356×10^{-6}	1.02	2.129×10^{-4}	1.00	158

Table 2: The GPU and CPU evaluation times per event, T_n^{GPU} and T_n^{CPU}, given as a function of the number n of gluons for $gg \to (n-2)\,g$ processes. The polynomial scaling measures are also shown, for the GPU, $P_n(3)$, and for the CPU, $P_n(4)$. The $P_n(m)$ are defined as $P_n(m) = [(n-1)/n] \times \sqrt[m]{T_n/T_{n-1}}$. The rightmost column finally displays the gain $G = T_n^{\text{CPU}}/T_n^{\text{GPU}}$.